Support of Asynchrony in Sensor Web

Liping Di, Genong (Eugene) Yu, Nengcheng Chen, Min Min, and Huilin Wang
Center for Spatial Information Science and Systems
George Mason University
6301 Ivy Lane, Suite 620
Greenbelt, MD 20770

Abstract- **Sensor Web is a system of systems, of which each system consist of live sensor, data, and geospatial processing services. In a broad sense, it includes archived data, and on-demand generated data or virtual sensor. In such a system, asynchrony cannot be avoided because of differences in clock/time and the long processing time of complex processes. Asynchrony can occur at the sensor planning stage, the data collection phase, or in the data processing process. Support of asynchrony is one of the core characteristics of the Coordination and Event Notification Services (CENS), the central component of the Self-Adaptive Earth Predictive System (SEPS) that operates Sensor Webs. This paper reviews the different types of asynchrony in the Sensor Web system and their technical solutions. Technologies in industry to support asynchronous mechanisms can be roughly grouped into two major categories: message queuing and publish-subscribe. Representative technologies include Web Services Addressing (WS-Addressing) for message queuing and the Extensible Messaging and Presence Protocol (XMPP) for publication-subscription. In emerging geospatial standards, the Web Notification Service (WNS) and the Sensor Alert Service (SAS) are designed to deal with asynchronies at message queuing pattern and publish-subscribe pattern correspondingly. There are generally two approaches in coordinating collaborative work by systems of systems. They are orchestration and choreography. Orchestration has a central director to guide the execution of each step such as a workflow. Choreography is a bottom-up approach, in which coordination is achieved by defining each individual web service/resource. In this project, orchestration is used. Business Process Execution Language is adopted as the script language to describe the composite processes. No matter which approach is chosen, a complex composite process involves syndication of all services and data as the final outcome. The syndication introduces problems similar to those, such as deadlock and reachability, encountered when trying to synchronize asynchronous threads in multi-thread programming. To solve these problems, proper methods for handling asynchrony in the Sensor Web system are crucial. With the acceptance of the Representation State Transfer (REST), the new style for Web Service, a new paradigm, called Resource Oriented Architecture (ROA), has emerged for interoperation in the Web environment. ROA is different from Service Oriented Architecture (SOA) in many aspects. Mechanisms for supporting asynchrony, e.g. HTTPEvents, are also emerging. Workflow standards specifically dealing with RESTful services, e.g. WfXML are emerging as well. Asynchronous prototypes for both resource-focused and service-focused workflows have been examined and demonstrated in two scenarios in this project. One is a wild fire workflow and another is the georeferencing workflow. Open research issues are also pointed out, especially those to be studied in the emerging Web 3.0.**

I. INTRODUCTION

Sensor Web is a system of systems. Each system consists of live sensor, data, and geospatial processing services. "A sensor web is a group of interoperable web services which all comply with a specific set of sensor behaviors and interfaces specifications"[1]. The definition of Sensor Web embraces not only observations directly acquired from physical sensors, but also derived products and historical data archives. Derived products can be computed on-demand and online through a series of Web services or processing services accessing real observations of sensors. These derived products can be called virtual geospatial products[2]. Integration and aggregation of component Web services in a loosely coupled Web environment form the crucial part of such a system. Geospatial products can be generated on-demand and customized to the requirements of users[2].

In the Sensor Web, asynchrony is required to allow the completion of long processing and the waiting of future observations. Asynchrony can occur at the sensor planning stage, the data collection phase, or in the data processing process. The following summarizes some of the typical properties of sensor Web systems.

(1) Observations required for assimilation and fusion of sensor data can be delayed [3]. For sensor Web, the observations may be available and planned for the future. It is not possible to let the Web service wait for days, weeks, or even months.

(2) Sensors in Sensor Web, no matter on board of satellite, aircraft, or in-situ, are mostly asynchronously event-driven systems[4]. Energy-awareness Sensor Web also requires asynchrony due to the delay of sensor hibernation period[5, 6]. The limited energy budget of sensors requires wireless sensors to have sleep and wake cycles.

(3) Virtual sensors, or models computed from geospatial resources and products may require Web services with long processing times.[7]. These geospatial Web Services may be asynchronous themselves to the non-blocking invocation from clients. The integrated services must be asynchronous.

In this project, asynchrony in the context of the Sensor Web has been studied. A general framework, Coordination and Event Notification Services (CENS), was developed to support different asynchronies for Sensor Web.

This paper is organized as follows. In Section II, asynchronous technologies for both geospatial Web services and other applications are reviewed. In Section III, CENS is introduced to harmonize the process of synchronous service. In Section IV, special handling techniques for integration of geospatial Web services and Sensor Web are studied. In Section V, two cases for applying the asynchronous framework and workflow are discussed. Finally, conclusions are presented and future research directions are described.

II. ASYNCHRONOUS TECHNOLOGIES

A. *Asynchrony in Web Services*

Web Services are computer software programs available on the Web. Most Web Services support synchronized access only. However, industrial applications did prompt the study of asynchrony since the default 60 second time-out for Web Services is not long enough for most processing algorithms. Client programs cannot wait longer than minutes for the server to respond in many cases.

1. Implementations and standards

Different strategies and technologies have been developed to support asynchrony. There are mainly two types of developments to support asynchronous Web services. One is asynchrony support at the transport level, for example, Java Message Service (JMS), Web Service Invocation Framework (WSIF), Java API for XML Messaging (JAXM), and Reliable HTTP (HTTPR)[8]. For implementing asynchronous Web services, Microsoft .NET and JAX-WS (Java API for XML Web Services) are two major commercial technologies.

Another strategy to support asynchrony is by developing standards or protocols. These may be realized through open frameworks, e.g. RosettaNet for industrial applications, IHE (Integrating the Healthcare Enterprise) for health study, and xCBL (XML Common Business Library) and ebXML (Electronic Business using eXtensible Markup Language) for business[8]. For Grid Web Services, OGSA (Open Grid Services Architecture) has developed support of asynchrony in its framework.

A Web Service, in its narrow definition of W3C, is a program that passes message through the Simple Object Access Protocol (SOAP), describes itself using the Web Service Description Language (WSDL), and advertises itself in a Universal Description Discovery and Integration (UDDI)[9]. To support asynchrony with the existing Web Service technology, especially SOAP, OASIS developed the Asynchronous Service Access Protocol (ASAP)[10]. ASAP was evolved from Internet Engineering Task Force (IETF) draft Simple Workflow Access Protocol (SWAP) that aims at providing a generic asynchronous service to create its instance, monitor its status, control its process, and notify its completion[10, 11]. ASAP also supersedes the Asynchronous Web Services Protocol (AWSP), which aims at solving the Web Service asynchrony problem through the use of SOAP[10, 12]. ASAP allows the creation of a generic asynchronous service instance, monitoring of its progress, and control of its execution[10].

REST (Representation State Transfer) is a "more constrained architectural style" [9] for Web applications. It has gained popularity because of its simplicity. W3C recognizes its existence and supports creation of REST Web services at both the message (e.g. SOAP 1.2) and description (e.g. WSDL 2.0) levels. To support and accommodate asynchrony in REST Web services, WfXML (XML Based Protocol for Run-Time Integration of Process Engines) was developed on top of ASAP to enable creation and monitoring of asynchronous instances[8, 13].

2. Asynchrony patterns

Asynchronies can be categorized into different groups depending on their invocation patterns. Several have reviewed and pointed out different existing patterns for asynchronous Web Services[14, 15].[16] Given that Sensor Web has been operated and developed mostly under service-oriented architecture, these asynchronous patterns are applicable to Sensor Web services.

Asynchronies can be handled at the client-side or at the server-side[17]. At the client side, two approaches, a nonblocking Application Programming Interface (API) and transport level, can be used[17]. The nonblocking API approach relies on a client side proxy service to deal with long server processing times. The transport level approach sends the request using one transport channel and receives the response using a different channel. At the server side, WS-Addressing can be used to handle asynchrony.

Client asynchrony patterns can include fire and forget, sync with server, poll object, and result callback[8, 14]. They can also use message queues[16, 18]. Based on their interaction patterns, asynchronies can be classified into Callback pattern, Publish/subscribe pattern, Polling pattern, Callback Factory Pattern, and Publish/Subscribe Factory Pattern[15]. For callback, two protocols can be adopted: WS-Callback[19] and WS-Addressing[20]. Details of these patterns are beyond the scope of this paper and can be found in [14-16, 18].

The Callback pattern is of special relevance to the Sensor Web since it frees the client from the heavy network traffic of polling between client proxy and server. Callback is the most widely supported pattern by industrial protocols, for example RosettaNet, xCBL, ebXML, IHE, and OGSA[15]. ASAP supports only the Callback Factory pattern. WS-Addressing is used for asynchronous SOAP geospatial Web services. The Publish/subscribe pattern is used in this system because it allows information to be distributed to a group of partners[15]. Publish/subscribe pattern is supported by ebXML and OGSA[15]. In this study, the ebXML protocol will be adopted and Extensible Messaging and Presence Protocol (XMPP) channel will be used to transmit the message for the Publish/subscribe pattern in Sensor Web[21].

B. *Geospatial Web Services*

Open Geospatial Consortium Sensor Web Enablement (OGC SWE) has developed a suite of specifications to enable "real time integration of heterogeneous sensor webs into the information infrastructure"[22]. Seven specifications have been developed under the OGC SWE: Observations & Measurements (O&M), Sensor Model Language (SensorML), Transducer Model Language (TransducerML), Sensor Observations Service (SOS), Sensor Planning Service (SPS), Sensor Alert Service (SAS), and Web Notification Service (WNS). O&M is the encoding schema for observations and measurements from sensors[23, 24]. SensorML encodes the descriptions for sensors and processes[25]. TransducerML is for encoding and streaming transducer data[26]. SOS is a service for managing observation data from sensors[27]. SPS is a service for sensor discovery and observation planning[28].

Asynchrony in the geospatial Sensor Web is supported by one specification, SAS, and one best practice, WNS. SAS allows the event notification through XMPP channel

publication (from the provider) and subscription (from the client)[29]. WNS is specifically designed to asynchronously deliver alerts/notifications from SAS and SPS[30].

III. CENS

In this project, WNS and SAS were used as the core foundation to support asynchrony for coordinating and harmonizing heterogeneous sensor webs and virtual sensors. A message notification approach was used to keep the final processes synchronized to complete complicated and/or lengthy geospatial processing workflows. Figure 1 shows the general framework for asynchronous process coordination and notification. CENS is the core of the large framework in developing SEPS (Self-Adaptive Earth Predictive Systems)[1]. Corresponding Web services will be notified of changes of state through multiple transport protocols, e.g. HTTP, email, telephone, and fax . Other modules of SEPS—Data Preprocessing, Integration, and Assimilation Services (PIAS), Data Discovery and Retrieval Services (DDRS), and Data and Sensor Planning Services (DSPS)—can be monitored and controlled using the asynchrony-enabled coordination system for discovering, downloading, and processing geospatial data and products.

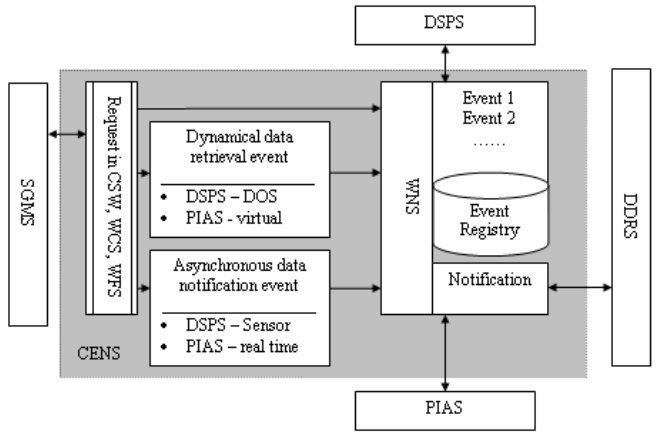

Figure 1. Architecture of CENS

IV. SERVICE INTEGRATION

From sensor observations to model input, the process is not trivial. Integration through re-use of geospatial Web services is one way to increase the efficiency of preparing and pre-processing raw observations for model input. To automate or assist the process of service integration, two approaches may be used – orchestration (top-down) and choreography (bottom-up). Orchestration has a central director to guide the execution of each step such as a workflow. Choreography is a bottom-up approach in which coordination is achieved by defining each individual Web service/resource. In this project, orchestration is used and Business Process Execution Language (BPEL) is adopted as the script language to describe the composite processes.

Support for asynchrony in workflows, requires the asynchronous invocation of individual Web services and the description of the workflow as an asynchronous Web service. WS-Addressing is one approach for supporting callback if a SOAP message is passed between web services. If the Web service uses WS-Addressing, the BPEL engine generates a proxy callback service to receive the response. Correlation between different services is established by using unique message identification in the WS-Addressing message ID.

V. ASYNCHRONOUS CASES

The CENS framework has been successfully applied in several cases. Here two scenarios are discussed: enabling the retrieving of subscription-based data and a live sensor planning system.

A. *Asynchronous access to data order system*

Geostationary Operational Environmental Satellites (GOES) data were required for weather prediction. They can be ordered from the Comprehensive Large Array-data Stewardship System (CLASS) of NOA. However, the NOAA CLASS is an order system. It requires user to submit a request through their Web pages and wait for an email notification before the data can be downloaded. This waiting period can vary, from half hour to hours. This cannot be done automatically using synchronous processing services/programs.

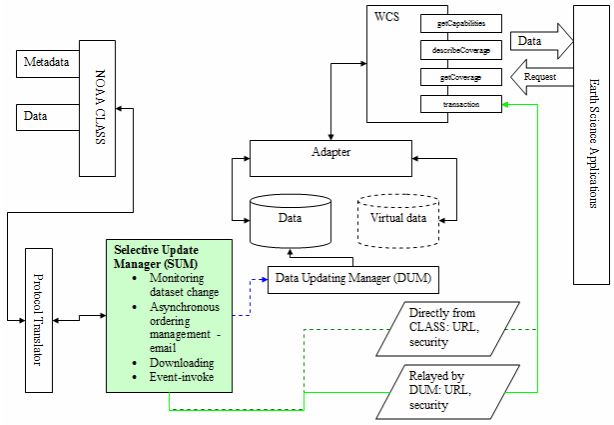

Figure 2. Asynchronous access to subscription-based archived sensor observations

Figure 2 shows the solution using the CENS framework. A parser was developed to parse the metadata through their NOAA CLASS and populate the data availability in the capability description files of WCS (Web Coverage Service). An adapter is used to monitor a given subscriber email account and generate a WNS notification message. When a user or a program requests the virtual data through the WCS, the WCS identifies the data is not actually hosted or cached locally. It sends out an order request to NOAA CLASS. The workflow is suspended. It waits for an email notification of processing status from the adapter service. Once the email notification is received, the downloading starts and the user or program is notified by WNS on the availability of data. By doing so, the network traffic load is reduced by queuing process events asynchronously.

B. *Georeferencing*

The Georeferencing case is a demonstration carried out in the OWS-5 of OGC interoperability campaign. This project implemented the workflow using standard BPEL and several

Web services: Web Processing Service (WPS), Sensor Observation Service (SOS),Sensor Alert Service (SAS),Sensor Planning Service (SPS), Web Coverage Service with enhancement of transaction capabilities (WCS-T),Jpeg Image Server (JPIP), and Catalog Service—Web Profile (CS/W) were chained together to plan geospatial data requests, retrieval, geo-referencing, and alerting. It was an international collaboration. The self-developed BPEL engine was used in executing the final Web service chain. Figure 3 shows the final workflow as displayed using the standard BPEL designer. In the case of designing the service chain, an Oracle BPEL designer was used to design the workflow.

Figure 3. BPEL workflow for geo-referencing from sensor observations

The workflow can be briefly described as: a user or a program *submit* planning request to the SPS. When an email notification is received on the confirmed acquisition of data, observation is retrieved from the SOS. It is then fed into the JPIP server through a secure transaction. The sensor description and the JPIP data are added to the WCS through its transaction operation. The data availability is made known through SAS to all subscribed users.

In this process, two types of asynchronies were used. One is for the first step of SPS based on WNS. Another is the final notification of data availability to all subscribed users through SAS. The first can be done through a WS-Addressing addition in the SOAP header part if the SOAP message protocol is adopted. It is a callback pattern asynchrony. The second one is a publish/subscribe pattern. Both were supported in the CENS through its core component BPEL engine.

VI. CONCLUSIONS

The asynchronous support in the SEPS has been made available through its core sub-system, CENS. BPEL is adopted as the core script language. An OGC-aware BPEL engine formed the core of the CENS. The framework and concepts of asynchrony for Sensor Web were applied in two scenarios. From the experiments, it can be observed that asynchrony cannot be avoided in Sensor Web, due to future observation planning and long processing time. Proper use of asynchrony would reduce the network traffic in some extent.

The CENS is at an early stage. Support for full OGC-specific asynchrony has not been completely studied and implemented. Performance evaluation should be quantitative. Simulation network systems may be used to assist on the experiments and analyses.

ACKNOWLEDGMENT

This study is funded by NASA AIST program (Grant # NNX06AG04G, PI: Dr. Liping Di). For the OWS-5 demonstration case, many helps had been received from many parties, including Mr. Shayne Urbanowski, Mr. Alexandre Robin, Mr. Peter Giacovelli, Mr. Steven Keens, Mr. Johannes Echterhoff, Mr. Max Martinez, Dr. Arne Broering, Dr. Peisheng Zhao, Dr. Yaxing Wei, and Dr. Weiguo Han.

REFERENCES

[1] L. Di, "Geospatial Sensor Web and Self-adaptive Earth Predictive Systems (SEPS)," in *ESTO/AIST Sensor Web PI Meeting, Febrary 13-14, 2007* San Diego, California, USA, 2006.

[2] L. Di, "Customizable Virtual Geospatial Products at Web/Grid Service Environment," in *IEEE International Geoscience and Remote Sensing Symposium (IGARSS '05). Proceedings*, Seoul, South Korea, 2005, pp. 4215- 4218.

[3] J. A. López-Orozco, J. M. d. l. Cruz, E. Besada, and P. Ruipérez, "An Asynchronous, Robust, and Distributed Multisensor Fusion System for Mobile Robots," *The International Journal of Robotics Research,* vol. 19, pp. 914-932, 2000.

[4] M. Yu, H. Mokhtar, and M. Merabti, "A Survey of Network Management Architecture in Wireless Sensor Network," in *The convergence of telecommunications, networking and broadcasting, 26-27 June 2006*, Liverpool, England, UK, 2006.

[5] C. I. Kelly, V. Ekanayake, and R. Manohar, "SNAP: A Sensor-Network Asynchronous Processor," in *Proceedings of the Ninth International Symposium on Asynchronous Circuits and Systems (ASYNC'03)*, 2003.

[6] H. S. Kim, T. F. Abdelzaher, and W. H. Kwon, "Minimum-Energy Asynchronous Dissemination to Mobile Sinks in Wireless Sensor Networks," in *ACM SenSys*, Los Angeles, California, USA, 2003.

[7] A. Joshi, "Active Web Alert Service for Rule Based Alerting in Sensor Web: An Event Based Approach." vol. M.S. Enschede, The Netherlands: International Insitute for Geo-Information Science and Earth Observation, 2005, p. 128.

[8] U. Zdun, M. Voelter, and M. Kircher, "Design and Implementation of an Asynchronous Invocation Framework forWeb Services," in *Web Services - ICWS-Europe 2003 (International Conference*

ICWS-Europe 2003 Erfurt, Germany, September 23-24, 2003 Proceedings). vol. 2853/2003, M. Jeckle and L.-J. Zhang, Eds.: Springer Berlin / Heidelberg, 2003, pp. 64-78.

[9] D. Booth, H. Haas, F. McCabe, E. Newcomer, M. Champion, C. Ferris, and D. Orchard, "Web Services Architecture," W3C, 2004.

[10] J. Fuller, M. Krishnan, K. Swenson, and J. Ricker, "Asynchronous Service Access Protocol (ASAP) Version 1.0 (wd-asap-spec-02f)," OASIS, 2006, p. 46.

[11] K. Swenson, "Simple Workflow Access Protocol (SWAP), IETF draft-swenson-swap-prot-00," IETF, 1998.

[12] K. Swenson and J. Ricker, "AWSP Asynchronous Web Services Protocol," 2002, p. 41.

[13] M. z. Muehlen, J. V. Nickerson, and K. D. Swenson, "Developing web services choreography standards—the case of REST vs. SOAP," *Decision Support Systems,* vol. 40, pp. 9-29, 2005.

[14] M. Voelter, M. Kircher, U. Zdun, and M. Englbrecht, "Patterns for Asynchronous Invocations in Distributed Object Frameworks," in *The 8th European Conference on Pattern Languages of Programs (EuroPlop 2003), June 2003*, Irsee, Germany, 2003, pp. 269-284.

[15] M. Brambilla, G. Guglielmetti, and C. Tziviskou, "Asynchronous Web Services Communication Patterns in Business Protocols (Proceedings of 6th International Conference on Web Information Systems Engineering, New York, NY, USA, November 20-22, 2005)," in *Web Information Systems Engineering – WISE 2005*. vol. 3806/2005, A. H. H. Ngu, M. Kitsuregawa, E. J. Neuhold, J.-Y. Chung, and Q. Z. Sheng, Eds.: Springer Berlin / Heidelberg, 2005, pp. 435-442.

[16] U. Zdun, M. Voelter, and M. Kircher, "Pattern-Based Design of an Asynchronous Invocation Framework for Web Services," *International Journal of Web Service Research,* vol. 1, pp. 1-14, 2004.

[17] E. Chinthaka, "Develop asynchronous Web services with Axis2," http://www.ibm.com/developerworks/library/ws-axis2/index.html: IBM, 2007.

[18] M. Voelter, M. Kircher, and U. Zdun, *Remoting Patterns: Foundations of Enterprise, Internet and Realtime Distributed Object Middleware*: Wiley, 2004.

[19] Y. Goland, M. Nottingham, and D. Orchard, "WS-CallBack Protocol (WS-CallBack) 0.91," BEA Systems Inc, 2003.

[20] D. Box, E. Christensen, F. Curbera, D. Ferguson, J. Frey, M. Hadley, C. Kaler, D. Langworthy, F. Leymann, B. Lovering, S. Lucco, S. Millet, N. Mukhi, M. Nottingham, D. Orchard, J. Shewchuk, E. Sindambiwe, T. Storey, S. Weerawarana, and S. Winkler, *Web Services Addressing (WS-Addressing)*: W3C, 2004.

[21] XMPP, "XMPP Protocols," http://www.xmpp.org/protocols/: XMPP, 2008.

[22] OGC, "Sensor Web Enablement WG." vol. 2008: Open Geospatial Consortium, 2008.

[23] S. Cox, "Observations and Measurements – Part 1 - Observation schema," OGC 07-022r1 ed: Open Geospatial Consortium, 2007, p. 85.

[24] S. Cox, "Observations and Measurements – Part 2 - Observation schema," OGC 07-002r3 ed: Open Geospatial Consortium, 2007, p. 46.

[25] M. Botts and A. Robin, "OpenGIS® Sensor Model Language (SensorML) Implementation Specification," OGC 07-000 ed: Open Geospatial Consortium, 2007, p. 180.

[26] S. Havens, "OpenGIS® Transducer Markup Language (TML) Implementation Specification," OGC 06-010r6 ed: Open Geospatial Consortium, 2007, p. 258.

[27] A. Na and M. Priest, "OpenGIS® Sensor Observation Service Implementation Specification," OGC 06-009r1 ed: Open Geospatial Consortium, 2006, p. 91.

[28] I. Simonis and P. C. Dibner, "OpenGIS® Sensor Planning Service Implementation Specification," OGC 07-014r3 ed: Open Geospatial Consortium, 2007, p. 186.

[29] I. Simonis and J. Echterhoff, "OGC® Sensor Alert Service Implementation Specification," OGC 06-028r5 ed: Open Geospatial Consortium, 2007, p. 144.

[30] I. Simonis and A. Wytzisk, "Draft OpenGIS® Web Notification Service Implementation Specification," OGC 06-095 ed, 2003, p. 64.